Thread-level parallelism

The importance of multiprocessors was growing throughout the 1990s as designers sought a way to build servers and supercomputers that achieved higher performance than a single microprocessor, while exploiting the tremendous cost-performance advantages of commodity microprocessors.

The slowdown in uniprocessor performance arising from diminishing returns in exploiting instruction-level parallelism combined with growing concern over power is leading to a new era in computer architecture—an era where multiprocessors play a major role from the low end to the high end.

This increased importance of multiprocessing reflects several factors:

- The dramatically lower efficiencies in silicon and energy use that were encountered between 2000 and 2005, as designers attempted to find and exploit more ILP.
- A growing interest in high-end servers, as cloud computing and Software-as-a-Service become more important.
- A growth in data-intensive application driven by the availability of massive amounts of data on the Internet.
- The insight that increasing performance on the desktop is less important, either because current performance is acceptable or because highly compute- and data-intensive applications are being carried out in the cloud.
- An improved understanding of how to use multiprocessors effectively, especially in server environments where there is significant natural parallelism.
- The advantages of leveraging a design investment by replication rather than unique design.

Here, we will focus on exploiting **thread-level parallelism** (TLP), which implies the existence of multiple program counters and hence is exploited primarily through MIMDs. Although MIMDs have been around for decades, the movement of thread-level parallelism to the forefront across the range of computing from embedded applications to high-end servers is relatively recent. Likewise, extensive use of TLP for general-purpose applications is relatively new.

Our focus throughout this section will be **multiprocessors**, which we define as computers consisting of tightly coupled processors whose coordination and usage are typically controlled by a single operating system and that share memory through a shared address space.

The multiprocessors we will examine typically range in size from a dual processor to dozens of processors and communicate through the sharing of memory. Although sharing through memory implies a shared address space, it does not necessarily mean there is a single physical memory. Such multiprocessors include both single-chip systems with multiple cores, known as **multicore**, and computers consisting of multiple chips, each of which may be a multicore design.

In addition to true multiprocessors, we will return to the topic of multithreading, a technique that supports multiple threads executing in an interleaved fashion on a single multiple-issue processor. Many multicore processors also include support for multithreading.